An Ontology-Based Infrastructure for Usability Design Patterns

Scott Henninger [1], Padmapriya Ashokkumar[1]

[1] Univ. of Nebraska-Lincoln, Computer Science and Eng., CC 0112,
Lincoln, NE 68588-0112 USA
{scotth, ashokkum}@unl.edu

Abstract. Usability patterns represent knowledge about known ways to design graphical user interfaces that are usable and meet the needs and expectations of users. There is currently a plethora of usability patterns published in books, private repositories and the World-Wide Web. The dominance of pattern creation and discovery efforts has neglected the emerging need to organize the patterns so they can become a proactive resource for developing interfaces. This paper presents an approach using Semantic Web concepts that turns informal patterns into formal representation capable of supporting systematic design methods. Through this method, loosely coupled pattern collections can be turned into strongly coupled pattern languages representing software patterns and the context in which usability patterns should be applied. Furthermore, we demonstrate how experience-based mechanisms can be utilized to continuously improve the resulting pattern libraries.

1 Usability Patterns as an Interface Design Resource

The development of interactive software systems, i.e. systems with significant user interface components, is currently faced with a dilemma. Design for usability is becoming increasingly important to the success of software systems, but software developers are usually poorly trained in human factors or usability issues. Education and iterative development processes aimed at evaluating and improving the user interfaces are necessary solutions to this problem, but techniques are needed that provide software developers with proactive knowledge and techniques for developing high quality user interfaces.

There is no lack of information and guidance on the design, development, and evaluation of user interfaces. Usability guidelines, patterns, principles, books, Web pages depicting good and bad examples, databases and various repositories are examples of both the plethora of knowledge and proliferation of formats that have been used to disseminate usability design knowledge. Design patterns, structured descriptions of a successful solution to a recurring problem, have emerged as an important tool that explicitly represents when and how a pattern should be used [7, 10].

Current approaches to representing patterns are document-based, at best supported with hypertext tools on the Web. These passive representations rely on individual developers to know of the existence of the resources and understand when they should be applied. Given the potentially copious numbers of patterns that can be used in different contexts and the lack of training in usability issues, this is not a satisfactory solution. Computational pattern representations are needed that facilitate retrieval, and support for design processes.

In addition, mechanisms are lacking to turn the currently isolated sets of patterns and pattern collections into a pattern language, [3] an interconnected corpus of knowledge that embodies a degree of consensus the design community. Tools and techniques are needed to create a community of practice in the patterns community that continuously evolves and refines the available usability patterns.

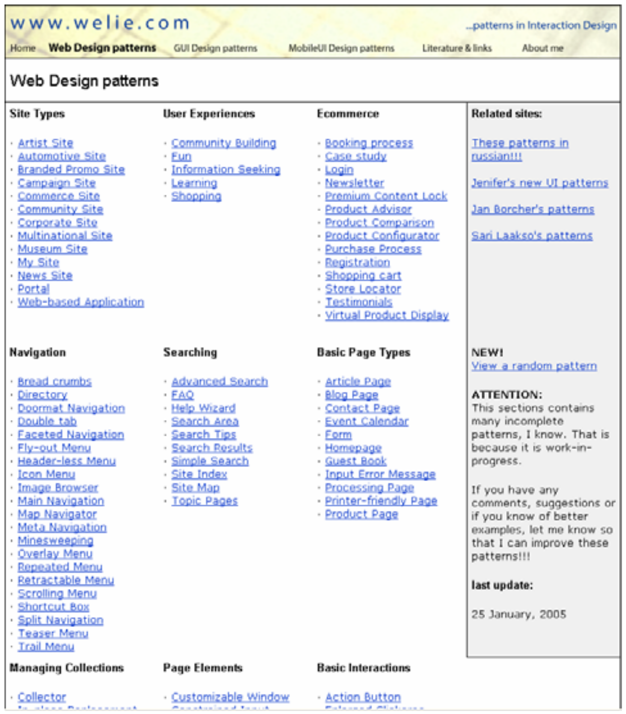
In this paper, we present a framework in which patterns are represented using Semantic Web technologies [4, 17] for creating ontologies. Ontologies are formal and explicitly defined specifications of concepts whose meaning is shared within a community [31]. We provide a specific example of ontologies and associated rules and inferences that allow intelligent support for applying usability patterns. This is demonstrated through a next-generation BORE (Building an Organizational Repository of Experiences) system, a software process framework tool for generating flexible development methodologies [21]. The BORE framework is extended and used to demonstrate how Semantic Web technologies can be utilized to support distributed ontologies and forms of computational reasoning to support the design of user interfaces. Description logic reasoning and rules are used to actively deliver patterns and other usability knowledge to the developer and find related patterns that fit the defined context of the application.

2 Usability Guidelines and Patterns

Usability guidelines have been used to disseminate usability knowledge and ensure a degree of consistency across applications [24, 29, 34]. Usability guidelines provide principles and concepts that lead to good interface design from both general and widget-specific perspectives. While hundreds of usability guidelines have been designed and published, the guideline statements tend to be stated in an abstract, decontextualized, manner. This leads to problems of interpretation, applicability, and ambiguous or contradictory statements [8, 32] that make it difficult for developers to properly apply usability guidelines.

A usability patterns community, inspired by the recent work on software patterns, [5] has begun to explore how patterns can be used to provide an intermediate perspective between universally applicable usability guidelines and component-specific style guides [25, 36]. The essential idea of a design pattern is to capture recurring problems along with the context and forces that operate on the problem to yield a general solution. Collections of patterns can be organized in a network of higher-level patterns that are resolved or refined by more detailed patterns, resulting in a *pattern language* [3].

Differences between usability guidelines and usability patterns lie primarily in perspective and representation of the information. The perspective of usability patterns tends to be more problem-oriented, focusing on describing a problem and solution,



Web Design patterns

Site Types

- Artist Site
- Automotive Site
- Branded Promo Site
- Campaign Site
- Commerce Site
- Community Site
- Corporate Site
- Multinational Site
- Museum Site
- My Site
- News Site
- Portal
- Web-based Application

User Experiences

- Community Building
- Fun
- Information Seeking
- Learning
- Shopping

Ecommerce

- Booking process
- Case study
- Login
- Newsletter
- Premium Content Lock
- Product Advisor
- Product Comparison
- Product Configurator
- Purchase Process
- Registration
- Shopping cart
- Store Locator
- Testimonials
- Virtual Product Display

Related sites:

These patterns in russian!!!

Jenifer's new UI patterns

Jan Borcher's patterns

Sari Laakso's patterns

Navigation

- Bread crumbs
- Directory
- Doormat Navigation
- Double tab
- Faceted Navigation
- Fly-out Menu
- Header-less Menu
- Icon Menu
- Image Browser
- Main Navigation
- Map Navigator
- Meta Navigation
- Minesweeping
- Overlay Menu
- Repeated Menu
- Retractable Menu
- Scrolling Menu
- Shortcut Box
- Split Navigation
- Teaser Menu
- Trail Menu

Searching

- Advanced Search
- FAQ
- Help Wizard
- Search Area
- Search Tips
- Search Results
- Simple Search
- Site Index
- Site Map
- Topic Pages

Basic Page Types

- Article Page
- Blog Page
- Contact Page
- Event Calendar
- Form
- Homepage
- Guest Book
- Input Error Message
- Processing Page
- Printer-friendly Page
- Product Page

NEW!
View a random pattern

ATTENTION:
This sections contains many incomplete patterns, I know. That is because it is work-in-progress.

If you have any comments, suggestions or if you know of better examples, let me know so that I can improve these patterns!!!

last update:

25 January, 2005

Managing Collections

- Collector

Page Elements

- Customizable Window

Basic Interactions

- Action Button

Fig. 1. An Online Usability Patterns Collection (from [van Welie 2005]).

than the more general information or advice perspective of guidelines. Although templates and data structures for describing guidelines and patterns can easily be reconciled, the 'context' and 'forces' fields commonly seen in pattern formats is indicative of a problem-oriented perspective

The goals of both these approaches are essentially the same: to document and manage experience about usability design issues in a format that is easily disseminated and understood. But current research has focused on the development of pattern and guideline content, with very little work being performed on tool support for applying the knowledge contained in guidelines and patterns.

2.1 Current Support for Usability Patterns

Suppose a project team is developing E-commerce website to serve users who want to purchase a set of products through a Web browser. The product offerings are large and diverse enough that it makes sense to divide the site into multiple Web pages with navigational aids to go between categories. But this leaves the sticky problem of how to collect items that have been chosen in different places in the site, both from a usability perspective and an information retention perspective (i.e. keeping track of chosen items across separate Web pages).

Some members of the team are aware that proven usability knowledge for these types of interfaces is available and refer to the Interaction Design Patterns website (often referred to as the Amsterdam Patterns Collection) [35] containing over 60 usability patterns (see Fig. 1), including guidelines relevant to the project such as Ecommerce and Web shopping patterns. Many other pattern collections exist, both in Web sites [15, 27, 33] and books, [34] and could also be used by the team instead of or in addition to this pattern collection.

Given the discovery of this pattern collection, the team must read, digest and sort out the collection of patterns to find which ones might be applicable to parts of their interface design. This leads to a number of problems when trying to design the system using the pattern collection. First, since the patterns are not represented in a problem-oriented form, it is not immediately clear which set of patterns apply to a particular problem. For example, the shopping cart pattern [35] is a solution to the problem of users purchasing items, but it is not clear which other patterns may also apply to this situation. The developers must read all the patterns and make decisions about the applicability of each pattern to the current project.

Second, after a particular pattern has been chosen, there are no indications or formal relationships about which pattern(s) should be used with the chosen pattern. For example, using the Shopping pattern may involve choices for specific interaction types, such as using a persistent button or frame to indicate and store persistent items across Web pages, or the need for specific types of search interactions. The patterns are represented in informal natural language, at best using hyper links, or a “Related patterns” field that link to other patterns in the collection. Otherwise, there is little to no information on how the patterns may work together for solutions to larger problems. In addition, there are no links between pattern collections, making the work for the development even harder, as the relationship between patterns in collections is not specified, or at best specified with a hypertext link that carries no semantics.

2.2 Tools for Finding Guidelines

Because usability patterns are such a new discipline, few or no tools have been built to help people find relevant patterns. Within the guidelines community, a number of tools have been developed, such as classification schemes, [28] and various information retrieval and hypertext tools [23]. Efforts have also defined mechanisms for validating guidelines, although few efforts have followed such standards. This is also the case for design patterns [14] where the tool most used are UML models and, but no attempt is made to investigate relationships between patterns and/or composing multiple patterns [2].

The potential utility of using the structured format of patterns, for example using the context field to formally or systematically indicate when a pattern should be used, has yet to be explored in any detail. The current state of affairs for pattern users are the existence of collections of patterns made available through a handful of portals [6, 9], books or Wikis. Even if there were HTML hypertext links between these collections (and there are very few), no semantic information is carried by a hyperlink, other than a on-dimensional and vague "is-related-to" (AKA "see-also", etc.).

A problem with all of these approaches is that they are based on passive repositories. People must know and understand that appropriate patterns exist before they will begin to look for them. Even then, people will have trouble articulating their queries in the language and mental models [13] used by pattern developers. Because of these issues passive repositories attached to search engines will always be vastly underutilized. Techniques that actively tell the user of potentially relevant information, such as critics, [11] and agents [17] are needed to make patterns a more effective tool for software developers.

3 Using the Semantic Web to Represent Pattern Languages

We have been investigating how currently defined Semantic Web standards (recommendations) can be utilized to define formal descriptions of usability patterns in a form that computers can understand and that can easily be converted into a human-readable form. In addition, the distributed facilities provided by the World-Wide Web raises possibilities for both tying multiple distributed pattern collections together while providing a computational medium that allows agents to make intelligent inferences on the relationships between patterns

3.1 Ontologies

In a computational context, an ontology is a description of the concepts and relationships between concepts that are formally defined within a domain of interest. Ontologies are created as a set of definitions from a formal vocabulary defining a "schema" and instances (often referred to as individuals) of the schema concepts. For example, Fig. 2 contains parts of a Protégé 3.0 [30] screen image using the Semantic Web language OWL (Web Ontology Language) that implements forms of Description Logics [26]. In the taxonomy (the "Asserted Hierarchy"), the concept 'Browsing' is defined as a type of 'GUI_Design'. Note that terms in this ontology have a class/subclass relationship which have been "Asserted", i.e. defined by an ontology designer.

In addition to supporting hierarchies and taxonomies, ontologies define concept properties, which are independently defined concepts that describe relationships between concepts. In Fig. 2, the concept Browsing has a number of properties (follow the red arrow), such as 'hasProblem', 'hasSolution', 'hasContext', 'hasAlternative', etc. Each of these properties is defined by a domain and a range. For example, the hasSolution property has a domain of Software_Pattern and a Range of PatternConcepts (not shown), meaning all individuals with the hasSolution property will define a relationship in which an instance of Software_Pattern "has solution" of type PatternConcepts.

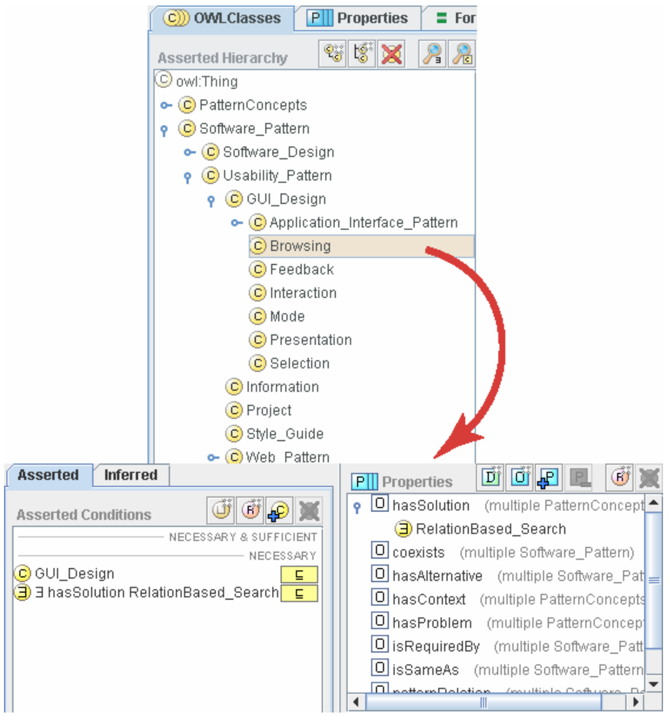
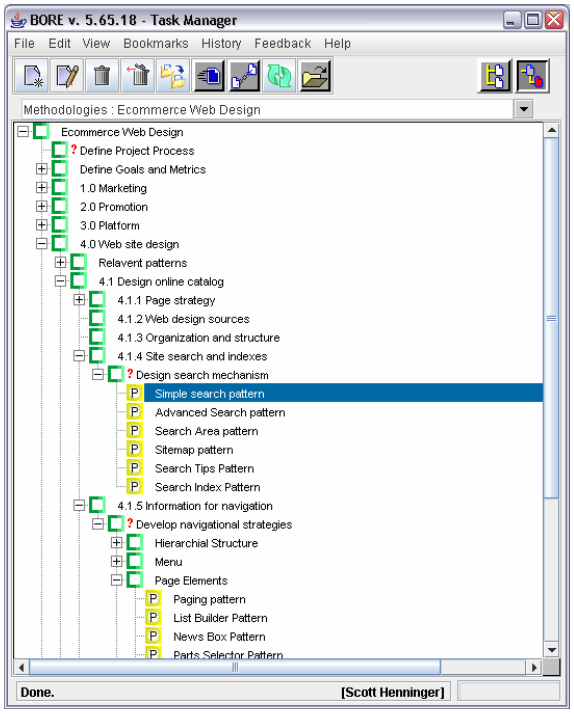


Fig. 2. An Example Class Structure of an Ontology for Usability.

In addition, the hasSolution property has an additional restriction denoted by the sub-property for hasSolution and the Necessary condition of the form "∃ hasSolution RelationBased_Search". '∃' is an existential quantifier that restricts values of 'hasSolution' to having at least one value from the class "RelationBased_Search". This means that any solution to the Browser pattern must have at least one solution from the class of RelationBased_Search concepts.

These are just a few examples of how OWL-DL can be used to formally describe usability patterns. Given this baseline, there are a number of classification and consistency checking inferences that can be made. For example, given an instance of the Browsing pattern with a hasSolution property with the individual treeViewBrowser, a reasoner can classify treeViewBrowser to be of type PatternConcept, given that RelationBased_Search is a subtype of PatternConcept (not shown). Furthermore, if there is only one hasSolution property, then it must be of type RelationBased_Search because of the existential quantification restriction. In this case, treeViewBrowser is classified as an instance of RelationBased_Search. Other examples of classification, as well as consistency checking using OWL will be demonstrated later in the paper.


BORE v. 5.65.18 - Task Manager
File
Edit
View
Bookmarks
History
Feedback
Help
Methodologies : Ecommerce Web Design
Done.
[Scott Henninger]

- Ecommerce Web Design
  - ? Define Project Process
  - Define Goals and Metrics
  - 1.0 Marketing
  - 2.0 Promotion
  - 3.0 Platform
  - 4.0 Web site design
    - Relevant patterns
    - 4.1 Design online catalog
      - 4.1.1 Page strategy
      - 4.1.2 Web design sources
      - 4.1.3 Organization and structure
      - 4.1.4 Site search and indexes
        - ? Design search mechanism
          - P Simple search pattern
          - P Advanced Search pattern
          - P Search Area pattern
          - P Sitemap pattern
          - P Search Tips Pattern
          - P Search Index Pattern
      - 4.1.5 Information for navigation
        - ? Develop navigational strategies
          - Hierarchial Structure
          - Menu
          - Page Elements
            - P Paging pattern
            - P List Builder Pattern
            - P News Box Pattern
            - P Parts Selector Pattern

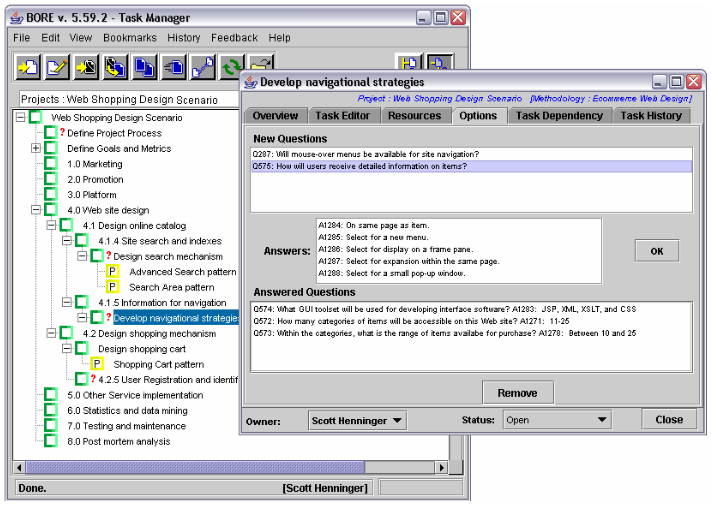
Fig. 3. The BORE Task Manager Showing a Methodology with

4 Building an Organizational Repository of Experiences

Instead of being solely relegated to a discretionary reference role, the knowledge contained in usability resources has the potential to be delivered as an integral part of the software development process. BORE (http://cse-ferg41.unl.edu/bore.html) is a process framework that organizations can use to create multiple methodologies that describe defined development processes [21]. BORE is used here as an existing framework for investigating new research in Semantic Web technologies and usability patterns. Previous work with BORE used an entirely different rule-based framework and explored the specific domain of software process tailoring [18, 21]. For this research, the hierarchical work breakdown structure and case organization of BORE is utilized to demonstrate how patterns are delivered to users, while replacing the internal system with a Semantic Web-based infrastructure.

4.1 Using BORE and Patterns

In a typical Web storefront, users traverse the Web site and find products to purchase. Lists of products are displayed and the user selects one or more. If the Web site is



Projects : Web Shopping Design Scenario

- Web Shopping Design Scenario
  - ? Define Project Process
  - Define Goals and Metrics
  - 1.0 Marketing
  - 2.0 Promotion
  - 3.0 Platform
  - 4.0 Web site design
    - 4.1 Design online catalog
      - 4.1.4 Site search and indexes
        - ? Design search mechanism
          - P Advanced Search pattern
          - P Search Area pattern
      - 4.1.5 Information for navigation
        - ? Develop navigational strategie
    - 4.2 Design shopping mechanism
      - Design shopping cart
        - P Shopping Cart pattern
      - ? 4.2.5 User Registration and identif
  - 5.0 Other Service implementation
  - 6.0 Statistics and data mining
  - 7.0 Testing and maintenance
  - 8.0 Post mortem analysis

Project : Web Shopping Design Scenario [Methodology : Ecommerce Web Design]

New Questions

Q287: Will mouse-over menus be available for site navigation?
Q575: How will users receive detailed information on items?

Answers:

A1284: On same page as item.
A1285: Select for a new menu.
A1286: Select for display on a frame pane.
A1287: Select for expansion within the same page.
A1288: Select for a small pop-up window.

Answered Questions

Q574: What GUI toolset will be used for developing interface software? A1283: JSP, XML, XSLT, and CSS
Q572: How many categories of items will be accessible on this Web site? A1271: 11-25
Q573: Within the categories, what is the range of items availabe for purchase? A1278: Between 10 and 25

Owner: Scott Henninger
Status: Open

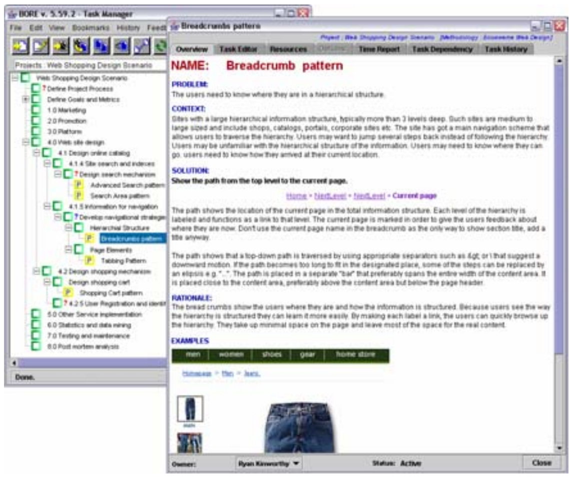
Fig. 4. Using Rule-Based Task Options in BORE.

complex enough, a shopping cart metaphor is used to allow people to browse between Web pages and retain items that have been purchased. Users/customers will often have the option of entering personal data for future sessions with the Web site.

Suppose a project team is developing such an e-commerce website using BORE. When the project is first created in BORE, an instance is created from a BORE methodology the "Ecommerce Web Design" methodology, whose work breakdown activities are shown in Fig. 3. E-commerce Web Design, with a set of activities. The activities in this methodology consist of all activities that have been defined for the domain in a manner similar to how the patterns were described in Owl in the previous section. Note that patterns are incorporated into the activities (patterns are denoted by the 'P' in the center of the activity icon) at specific places in the process. This is done because patterns can be seen as major activities that a project is held responsible for. Not all of the activities shown in Fig. 3 for the Ecommerce Web design methodology will be used by any one project. Each project will customize the methodology to their needs using the methodology rules.

4.2 Defining Context Through Task Options

Tailoring the methodology to meet the needs of a project is a major task addressed by BORE [21]. This is accomplished through options defined within the activity structure using Question/Answer pairs (see Fig. 4). These options can be defined on any activity in the methodology and are normally used to break an activity down into constituent sub-activities. Those with options are denoted in the Task Manager with a red '?', as seen in Fig. 3 and Fig. 4. Choosing the options define project characteristics (a representation of functional and non-functional requirements) that define the


BORE v. 5.59.2 - Task Manager
File Edit View Bookmarks History Feedb
Projects : Web Shopping Design Scenario
Web Shopping Design Scenario
? Define Project Process
Define Goals and Metrics
1.0 Marketing
2.0 Promotion
3.0 Platform
4.0 Web site design
4.1 Design online catalog
4.1.4 Site search and indexes
? Design search mechanism
Advanced Search pattern
Search Area pattern
4.1.5 Information for navigation
? Develop navigational strategies
Hierarchial Structure
Breadcrumbs pattern
Page Elements
Tabbing Pattern
4.2 Design shopping mechanism
Design shopping cart
Shopping Cart pattern
? 4.2.5 User Registration and identif
5.0 Other Service implementation
6.0 Statistics and data mining
7.0 Testing and maintenance
8.0 Post mortem analysis
Done.
Breadcrumbs pattern
Project : Web Shopping Design Scenario [Methodology : Ecommerce Web Design]
Overview
Task Editor
Resources
Time Report
Task Dependency
Task History
men
women
shoes
gear
home store
Homepage > Men > Jeans
Owner:
Ryan Kenworthy
Status: Active
Close

NAME: Breadcrumb pattern

PROBLEM:
The users need to know where they are in a hierarchical structure.

CONTEXT:
Sites with a large hierarchical information structure, typically more than 3 levels deep. Such sites are medium to large sized and include shops, catalogs, portals, corporate sites etc. The site has got a main navigation scheme that allows users to traverse the hierarchy. Users may want to jump several steps back instead of following the hierarchy. Users may be unfamiliar with the hierarchical structure of the information. Users may need to know where they can go. users need to know how they arrived at their current location.

SOLUTION:
Show the path from the top level to the current page.

Home > NextLevel > NextLevel > **Current page**

The path shows the location of the current page in the total information structure. Each level of the hierarchy is labeled and functions as a link to that level. The current page is marked in order to give the users feedback about where they are now. Don't use the current page name in the breadcrumb as the only way to show section title, add a title anyway.

The path shows that a top-down path is traversed by using appropriate separators such as &gt; or \ that suggest a downward motion. If the path becomes too long to fit in the designated place, some of the steps can be replaced by an elipsis e.g. "...". The path is placed in a separate "bar" that preferably spans the entire width of the content area. It is placed close to the content area, preferably above the content area but below the page header.

RATIONALE:
The bread crumbs show the users where they are and how the information is structured. Because users see the way the hierarchy is structured they can learn it more easily. By making each label a link, the users can quickly browse up the hierarchy. They take up minimal space on the page and leave most of the space for the real content.

EXAMPLES

Fig. 5. A Usability Pattern in BORE.

special needs of the project. In the background, rules fire when pre-specified Q/A pairs are chosen.

For example, when the '11-25' is chosen for the question "How many categories of items will be accessible on this Web site?" **and** "Between 10 and 25" is chosen for "Within the categories, what is the range of items available for purchase", then the "Shopping Cart pattern", amongst other activities, is chosen as a means to address a domain with a fairly high number of categories and categories per item.

Rules in BORE use a different mechanism than the OWL inferencing described in previous sections. Currently a proprietary forward-chaining rule engine is employed to define preconditions, actions and fire actions with backtracking options [19]. Later versions will use W3C SWRL (Semantic Web Rule Language) [22] tools, once the SWRL recommendation is finalized.

Depending on the user's questions and answers, many other patterns may be included in the workflow of this project. One of the main patterns for the Ecommerce Web Design domain is the Shopping Cart pattern. When a pattern is double-clicked, its corresponding OWL file, which is based on XML, is parsed, and converted into HTML for display in the case window, as shown in Fig. 5.

4.3 Inferring Relationships Among Patterns

Fig. 6 shows parts of an ontology that begins to implement a pattern language for the online shopping domain. Note that the "Asserted Hierarchy" view (Class Hierarchy, Callout 1) stems from the "Pattern_Level" root concept, which is modeled after a usability pattern language [37]. The *Shopping* pattern, being a subclass of *ExperienceLevel* is an "experience" level pattern. Experience level Patterns describe common experiences for which lower level patterns like Task Level Patterns and Action

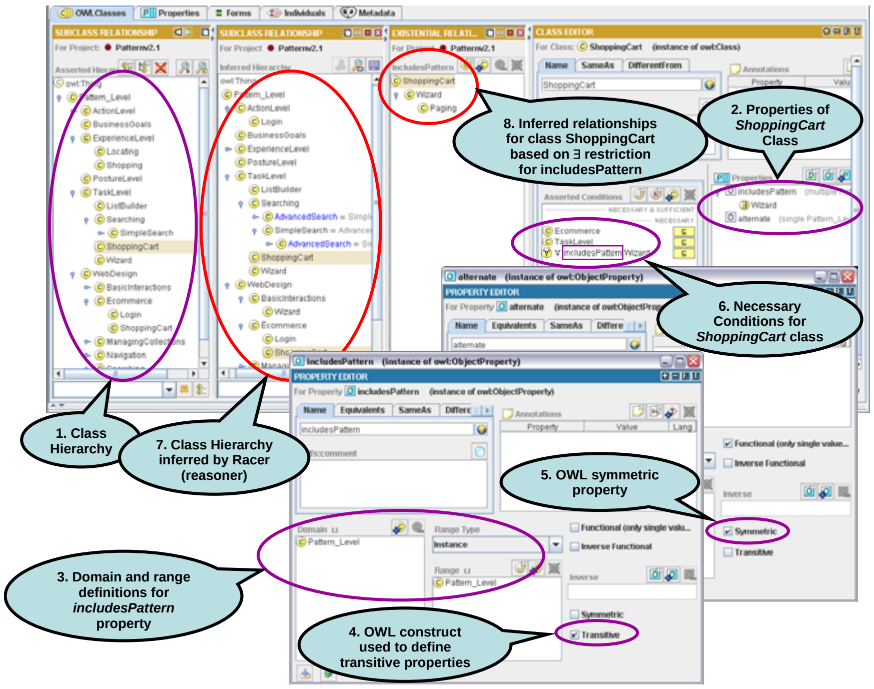

1. Class Hierarchy
2. Properties of *ShoppingCart* Class
3. Domain and range definitions for *includesPattern* property
4. OWL construct used to define transitive properties
5. OWL symmetric property
6. Necessary Conditions for *ShoppingCart* class
7. Class Hierarchy inferred by Racer (reasoner)
8. Inferred relationships for class ShoppingCart based on ∃ restriction for includesPattern

Fig. 6. An Example Pattern Language shown in Protégé.

Level Patterns can be used to create that experience. Task Level patterns such as *ShoppingCart*, *ProductComparison* perform a task like choosing products to buy or comparing products. Task Level Patterns are linked to Experience Level Patterns using various relationships. Similarly Action Level patterns are linked to Task Level patterns. The concepts in this ontology represent patterns such as *ShoppingCart*, *Wizard*, *Paging* etc. (see callout 1 in Fig. 6). *ShoppingCart* is a subclass of both the *Ecommerce* pattern concept and the *TaskLevel* pattern concept.

The properties (see Callout 2 in Fig. 6) represent relationships between instances of patterns. Some example relationships between patterns in this ontology include: Pattern A *contains* Pattern B, Pattern A is *equivalent* to Pattern B, Pattern A is an *alternate* to Pattern B. Pattern A is *specialization* of Pattern B, Pattern A is *to be used in combination with* either Pattern B or Pattern C, pattern A is *disjoint with* pattern B. These are a few examples of relationships that exist in our ontology, but are not explained here for the sake of conciseness. Another example, shown in Callout 2 of Fig. 6, is the *includesPattern* property defined with the domain and range defined as the type Pattern_Level (see Callout 3 in Fig. 6). The *includesPattern* property is also transitive (Callout 4), meaning that Reasoners can infer transitivity relationships with other class instances defined with the *includesPattern* properties.

Callout 5 shows that the *alternate* property is symmetric, meaning that an individual **a** related to an individual **b** by *alternate* also defines a relationship from individual **b** to **a** of the same type, *alternate*. For our ontology, this means that if patternA is an alternate for patternB, then patternB is also an alternate pattern for patternA.

Necessary Conditions, see Callout 6 in Fig. 6, define the conditions that must necessarily be fulfilled to be a member of the class. In this example, an individual must be a subclass of *Ecommerce* **and** a subclass of *TaskLevel*. In addition, the includesPattern property is restricted to members of the *Wizard* class through a universal quantifier (∀). This means that all included patterns (through the *includesPattern* property) must be a type of *Wizard*. The individual may have other attributes and classifications as well, but as long as the Necessary Conditions are fulfilled, the individual will be classified as a *ShoppingCart* pattern.

Areas marked in red in Fig. 6 represent inferences made with the help of a reasoner[1] and OWL constructs. Callout 7 shows the inferred class hierarchy. Note that *AdvancedSearc*h has been inferred to be a type of *Searching* that is equivalent to the class *SimpleSearch*. This occurs because *SimpleSearch* definesa restriction on the *alternate* property that it must be a type of *AdvancedSearch*. Since *alternate* is a symmetric property, then *AdvancedSarch* has the same alternate property. I.e. *AdvancedSarch* has the (rather strange) property restriction "∃ alternate AdvancedSearch". The only difference between the classes is that AdvancedSearch is a type of SimpleSearch and SimpleSearch is a type of Searching. Since both are subclasses of Searching (subtype is by definition a transitive relationship), the class definitions are equivalent. This inference is a good example of the kind of consistency checking that reasoners can do with OWL description logic constructs.

Another example of inferencing is shown in Callout 8, which shows the inferred relationships for the pattern *ShoppingCart* based on property *includesPattern*. Since *includesPattern* is a transitive relationship and *ShoppingCart includesPattern Wizard* and *Wizard includesPattern Paging*, the inferred relationships show both *Wizard* and *Paging* as subclasses of *ShoppingCart*. This is equivalent to saying "When you use patternA, you must use patternB" or "if you want to use patternA, then paternB might be an alternative".

Thus when a developer selects a specific pattern to use, the reasoner processes the relationships of that pattern with other patterns in the knowledge base and infers and returns all other related patterns that the developer might not be aware of, based on the relationships that have been defined in the ontology. Similarly when two patterns are alternatives to each other the developer may choose to use either of them. Based on the pattern chosen, other patterns are included because of the relationships defined in the ontology. This provides information to BORE or other design tool that help people to rapidly explore and choose design alternatives.

For example, whenever a searching mechanism like *SimpleSearch* or *AdvancedSearch* is used also include *SearchTips* pattern. This kind of domain knowledge can be represented as if-then rules and applied using a rule engine. Whenever the *if* clause is satisfied, the actions in the *then* part are performed.

The main advantage of using a Semantic Web tool like BORE is that the two most time consuming and complicated steps of this process are taken care of by the tool itself. Previously, the team had to become pattern experts themselves in order to fully

[1] For this example, we used the Racer [16] plug-in for Protégé.

understand all the patterns in the repository. With BORE, the team gets the same benefit of the patterns, but without having to become pattern experts or determine relationships between patterns.

4.4 Knowledge Building vs. Knowledge Management

The BORE tool, in combination with Semantic Web technology, is not just a repository and an intelligent search engine, but a means to collectively draw on the experiences of a wide range of usability projects and the different contexts that led to different decisions and designs. The formal, but flexible, nature of the Semantic Web supports the role of learning and creating new knowledge in the creative process of design [12].

Thus, knowledge is built, not managed in the process of design [20]. The inheritance hierarchy of the Semantic Web allows the flexible definition of concepts and fine-grained distinctions between them. The contextual and relationships attributes of the classes allow the definition of a context-sensitive pattern language capable of telling developers when and why a particular pattern should be used in a specific context. The context of the design problem can be represented in many ways, such as user-task models. BORE uses a rule-based representation that explicitly captures the requirements of the system.

5 Contributions and Future Work

The general goals of this approach are threefold: 1) To build a tool that puts context-appropriate usability guidelines at the fingertips of software designers and usability specialists so they can be used early in the design and development process. 2) To create a formal framework for creating an interconnected pattern language for usability knowledge. 3) To construct tools to facilitate community-based evolution of knowledge in the usability community through the combination of design patterns and Semantic Web representations.

In doing so, we have begun to resolve many of the problems that currently plague the usability patterns community, as well as the software patterns community as a whole. While a main goal of patterns is to form a vocabulary that helps developers communicate better [38], too many pattern collections have been created that draw little or no relationships between each other, in essence creating islands of patterns that sometimes contradict, duplicate, or are inconsistent with one another.

We see context as one of the main organizing features of patterns. Usability issues and decisions are often, if not always, context-sensitive. Capturing this context is just the first step at making usability design a more stable or scientific endeavor.

Beyond the critical need for evaluating various aspect of our ontology-based pattern language approach, there is a need to port the current BORE prototype from its currently fractured state into a more integrated and accessible platform. We are currently focusing on the Eclipse platform as a framework for this integration effort. This has the simultaneous advantage of being an increasingly accepted platform for development and the focus of current efforts to integrate Semantic Web tools and technologies, such as the Protégé ontology tool used in this work, into Eclipse plug-ins.

6 Conclusions

At the end of our research we end up with two related, but distinct, entities 1) an open environment for the creation and validation of usability patterns and relationships between them using intelligent Semantic Web technology, and 2) the BORE tool that uses this structure to deliver usability knowledge to software developers that may or may not have had any usability training.

The objective of this research is not an attempt to automate user interface design. To the contrary, it is recognized that effective user interface design take a degree of talent and careful work with the end users that cannot be captured through rules, patterns or any information system. Nonetheless, there is recognized knowledge and conventions that can help some designers reach higher levels of competency and help accomplished designers extend their knowledge to areas they have not yet experienced. This research is an exploration of how resources can be delivered to software developers through a representational medium that serves to establish relationships between context and usability resources and serve as a formal mechanism for communicating and refining usability design knowledge.

Continued research is needed to further understand the complexities of creating repositories of usability patterns and applying them proactively in the software development process. We have only taken small steps in this direction, and hope that future validation and use of our approach provide more information of usability knowledge and the contextual factors that impact this knowledge.

Acknowledgements. We gratefully acknowledge the efforts a number of students that have contributed in various ways in this and associated research, including Ryan Kinworthy, Rishi Kumar, Sarita Navuluru, and Sarathkumar Polireddy. This research was funded by the National Science Foundation (CCR-9988540, and ITR/SEL-0085788).

References

1. M. S. Ackerman, T. W. Malone, "Answer Garden: A tool for growing organizational memory," Proceedings of the Conference on Office Information Systems, ACM, New York, pp. 31-39, 1990.
2. C. Alexander, "The Origins of Pattern Theory: the Future of the Theory, and the Generation of a Living World," OOPSLA 1996 Keynote Address, http://www.patternlanguage.com/archive/ieee/ieeetext.htm, 1996.
3. C. Alexander, The Timeless Way of Building. Oxford Univ. Press, New York, 1979.
4. T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web," in Scientific American, vol. May 2001, 2001.

5. J. Borchers, "CHI Meets PLoP: An Interaction Patterns Workshop," in SIGCHI Bulletin, vol. 32, 2000, pp. 9-12.
6. J. Borchers, "hcipatterns.org: Patterns," http://www.hcipatterns.org/patterns.html, Last Updated March 2004, 2004.
7. G. Casaday, "Notes on a Pattern Language for Interactive Usability," Proc. Human Factors in Computing Systems (CHI '97), Atlanta, GA, ACM, pp. 289-290, 1997.
8. F. de Souza, N. Bevan, "The Use of Guidelines in Menu Interface Design: Evaluation of a Draft Standard," Human-Computer Interaction - INTERACT '90, Elsevier, North-Holland, pp. 435-440, 1990.
9. T. Erickson, "The Interaction Design Patterns Page," http://www.visi.com/~snowfall/InteractionPatterns.html, Last Updated March 2005, 2005.
10. T. Erickson, "Lingua Francas for Design: Sacred Places and Pattern Languages," Proc. Designing Interactive Systems (DIS 2000), New York, pp. 357-368, 2000.
11. G. Fischer, A. C. Lemke, T. Mastaglio, A. I. Morch, "Critics: An Emerging Approach to Knowledge-Based Human Computer Interaction," in International Journal of Man-Machine Studies, vol. 35, 1991, pp. 695-721.
12. G. Fischer, J. Ostwald, "Knowledge Management: Problems, Promises, Realities, and Challenges," in IEEE Intelligent Systems, vol. 16, 2001, pp. 60-72.
13. G. W. Furnas, T. K. Landauer, L. M. Gomez, S. T. Dumais, "The Vocabulary Problem in Human-System Communication," in Communications of the ACM, vol. 30, 1987, pp. 964-971.
14. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1995.
15. R. Griffiths, "The Brighton Usability Pattern Collection," http://www.cmis.brighton.ac.uk/research/patterns/home.html, 2002.
16. V. Haarslev, R. Möller, "Racer: An OWL Reasoning Agent for the Semantic Web," Proc. Int'l Wkshp on Applications, Products and Services of Web-based Support Systems (Held at 2003 IEEE/WIC Int'l Conf. on Web Intelligence), Halifax, Canada, pp. 91-95, 2003.
17. J. Hendler, "Agents and the Semantic Web," in IEEE Intelligent Systems, vol. 16, 2001, pp. 30-37.
18. S. Henninger, "Tool Support for Experience-Based Methodologies," in Advances in Learning Software Organizations (LSO 2002 Revised Papers), vol. LNCS 2640, 2003, pp. 44-59.
19. S. Henninger, "Tool Support for Experience-Based Software Development Methodologies," in Advances in Computing, vol. 59, 2003, pp. 29-82.
20. S. Henninger, "Turning Development Standards Into Repositories of Experiences," in Software Process Improvement and Practice, vol. 6, 2001, pp. 141-155.
21. S. Henninger, A. Ivaturi, K. Nuli, A. Thirunavukkaras, "Supporting Adaptable Methodologies to Meet Evolving Project Needs," Joint Conference on XP Universe and Agile Universe, Chicago, IL, pp. 33-44, 2002.
22. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C, http://www.w3.org/Submission/SWRL/, Last Updated May 21, 2004.
23. R. Iannella, "HyperSAM: A Practical User Interface Guidelines Management System," Proceedings of the Second Annual CHISIG (Queensland) Symposium - QCHI '94, Bond Univ., Australia, 1994.
24. S. J. Koyanl, R. W. Bailey, J. R. Nall, "Research-Based Web Design & Usability Guidelines," Communications Technology Branch, National Cancer Institute & US Dept of health and Human Services, http://www.usability.gov/pdfs/guidelines.html, 2003.
25. M. J. Mahemoff, L. J. Johnston, "Principles for a Usability-Oriented Pattern Language," Proc. Australian Computer Human Interaction Conference OZCJI 98, Adelaide, IEEE Computer society Press, pp. 132-139, 1998.

26. E. Miller, R. Swick, D. Brickley, B. McBride, J. Hendler, G. Schreiber, D. Connolly, "W3C Semantic Web," World-Wide Web Consortium, http://www.w3.org/2001/sw/, Last Updated Aug. 19, 2005.
27. PoInter, "Patterns of Interaction: a Pattern Language for CSCW," http://www.comp.lancs.ac.uk/computing/research/cseg/projects/pointer/pointer.html, Accessed: June 2005, 2005.
28. D. Scapin, C. Leulier, J. Vanderdonckt, C. Mariage, C. Bastien, C. Farenc, P. Palanque, R. Bastide, "A Framework for Organizing Web Usability Guidelines," 6th Conf. on Human Factors and the Web, Austin, TX, on-line at: http://www.tri.sbc.com/hfweb/scapin/Scapin.html, 2000.
29. S. L. Smith, J. N. Mosier, "Guidelines for Designing User Interface Software," ESD-TR-86-278, Technical Report, The MITRE Corporation, 1986.
30. Stanford Univ., "Protégé Project," National Library of Medicine, http://protege.stanford.edu/, Last Updated Feb. 2005, 2005.
31. R. Studer, V. R. Benjamins, D. Fensel, "Knowledge Engineering: Principles and Methods," in Data and Knowledge Engineering, vol. 25, 1998, pp. 161-197.
32. L. Tetzlaff, D. R. Schwartz, "The Use of Guidelines in Interface Design," Proc. Human Factors in Computing Systems (CHI '91), ACM, New York, pp. 329-333, 1991.
33. J. Tidwell, "UI Patterns and Techniques," http://time-tripper.com/uipatterns/, Last Updated May, 2005.
34. D. K. van Duyne, J. A. Landay, J. I. Hong, The Design Of Sites. Addison-Wesley, 2002.
35. M. van Welie, "Patterns in Interaction Design," http://www.welie.com/, Last Updated 25-05-2005, 2005.
36. M. van Welie, G. van der Veer, A. Eliens, "Patterns as Tools for User Interface Design," Workshop on Tools for Working With Guidelines, Biarritz, France, 2000.
37. M. van Welie, G. C. van der Veer, "Pattern Languages in Interaction Design: Structure and Organization," Proceedings of Interact '03, Zürich, Switserland, M. Rauterberg, Wesson, Ed(s). IOS Press, Amsterdam, The Netherlands, pp. 527-534, 2003.
38. J. Vlissides, "Patterns, The Top 10 Misconceptions," in Object Magazine, 1997.